Session 134

Building AppleScript Applications

Chris Espinosa
Manager, Components & Scripting

Introduction

- Nuts-and-bolts of developing scriptable applications and OSA client applications for Mac OS X
- Focused on Project Builder and Carbon
- Special issues raised by Mac OS X

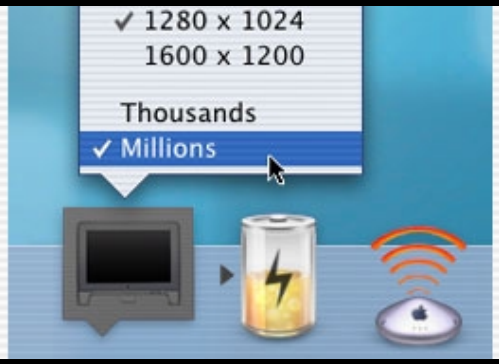
What You'll Learn

- How to port your scriptable apps to Mac OS X
- What to do about the FSSpec problem
- How to deal with new Mac OS X constructs like sheets
- How to debug scriptable applications and scripts

Developing With Project Builder

- Include a terminology resource
- Set the NSAppleScriptEnabled entry in your info.plist file

Demo



Debugging With gdb

- Looking at four-character codes
- Finding where handlers are
- Breaking on event dispatch to look at the event and its reply
- Looking at an Apple event or AEDesc

Working With OSTypes

- Showing a four-character code in gdb:
  print (char[4]) desc.descriptorType
- Setting a variable to a four-character code:
  set desc.descriptorType = *(long *)"TEXT"

Breaking on Dispatch

- **break AEMDispatch**
  - Breaks with event in r3
- **break InvokeAEEventHandlerUPP**
  - Breaks with handler address in r6
  - To break in that handler, **break *$r6**

gdb Scripts

- Write short scripts using gdb commands
- Call **malloc()** to get temporary memory
- Call Apple event routines to get information
- Save scripts in home-directory files
- Use **source** command in your **~/.gdbinit** file to load at gdb startup time

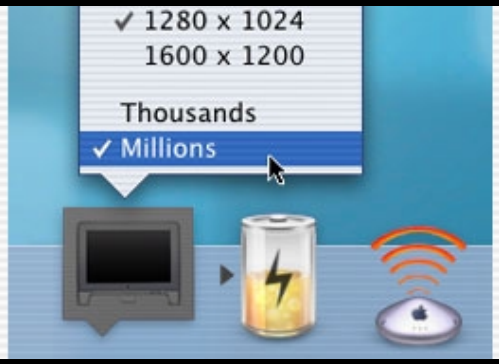
Printing an Event or Desc

```
define aedesc
    call (void *) malloc(4)
    set $handle = $
    call (long) AEPrintDescToHandle($arg0, $handle)
    if $ == 0
        printf "desc @ %p = {\n type = ‘%.4s’\n,
            storage (%p) = %s\n}\n", $arg0, $arg0,
            (long *)&$arg0[1], **(char ***) $handle
    end
    call (void) DisposeHandle(*$handle)
    call (void) free($handle)
end
```

Getting an Installed Handler

```
define gethandler
    call (void *) malloc(8)
    set $buf = $
    call (long) AEGetEventHandler($arg0, $arg1, $buf,
        $buf+4, 0)
    if $ == 0
        echo "found at "
        x /1wx $buf
    else
        echo "not found.\n "
    end
    call (void) free($buf)
end
```

Demo



Special Topics for Mac OS X

- Change in Apple event handler prototype
- The Death of **typeFSSpec**
- Handling Sheets
- Loading Scripting Additions

Apple Event Prototype Change

- Historically, refcons have been been signed longs
- For some reason, Apple event handler refcons were unsigned
- We changed this to avoid lots of casts
- You'll have to change your sources

The Death of **typeFSSpec**

- FSSpecs can't refer to long or Unicode file names
- FSRefs can't refer to nonexistent files
- Neither is valid across boxes or processes
- What do you do?

The Death of typeFSSpec

- Use typeAlias for file references in Apple events whenever possible
  - Valid across processes
  - Will represent long Unicode file names
  - Can distinguish between same-name volumes

The Death of **typeFSSpec**

- For nonexistent files, **use typeFileURL**:
  - Same data format as the **furl** drag flavor
  - Can refer to nonexistent files (unlike **typeAlias** and **typeFSRef**)
  - Encodes special characters in directory names ('/', ':', Unicode characters)
  - Can distinguish between same-named volumes
  - Is valid across processes, boxes, and even machines

The Death of **typeFSSpec**

- Sample code to create URL strings
  - **FURLCreateCFURLFromNavReply**
  - **FURLPutKeyCFURL** (and friends)
- Sample code to get and use URL strings
  - **FURLGetKeyCFURL** (and friends)
  - **FURLCreateFileWithCFURL**
- Coercions to/from **typeFileURL**
  - **typeChar**, **typeStyledText**, **typeUnicodeText**
  - **cFile**, **typeFSS**, **typeFSRef**, **typeAlias**
  - **typeObjectSpecifier**

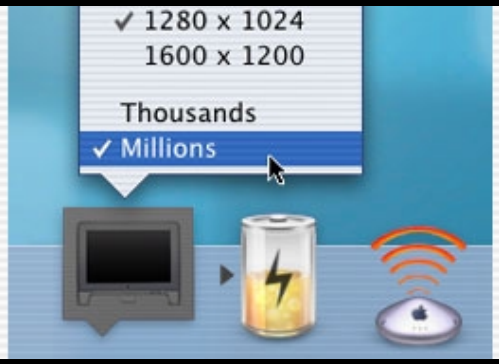
Example: Processing a NavReplyRecord

```
CFURLRef url;
err = FURLCreateCFURLFromNavReply(
    kCFAllocatorDefault, &theNavReply, &url);
        if (url && err == noErr) {
            err = FURLPutParamCFURL(&saveEvent,
                keyDirectObject, url);
            CFRelease(url);
        }
```

Example: Getting an **in** Parameter

```
err = AEGetParamPtr(ev, keyAEFile, typeFSRef, &t,
    &ref, sizeof(FSRef), &s);
if (err == fnfErr) {
    CFURLRef url;
    err = FURLCreateFromAppleEventParam(
        kCFAllocatorDefault, ev, keyAEFile, &url);
    if (url && err == noErr) {
        err = FURLCreateWithCFURL(url, &ref);
        CFRelease(url);
    }
}
```

Demo



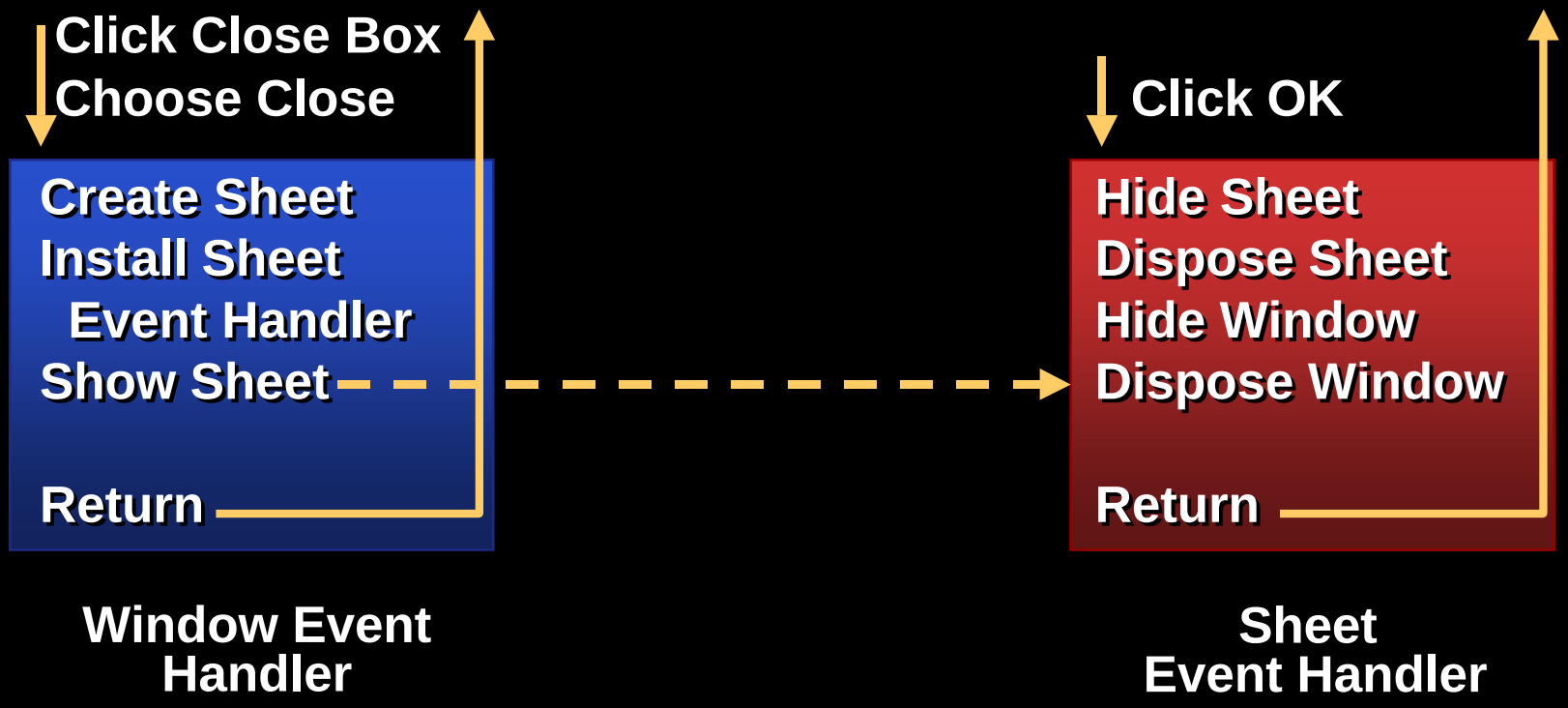
The Death of **typeFSSpec**

- What this means to you
  - Remove all **typeFSSpec** parameters and properties from your Carbon application
  - Use **typeFileURL** where you specifically support nonexistent files
  - Use **typeAlias** everywhere else

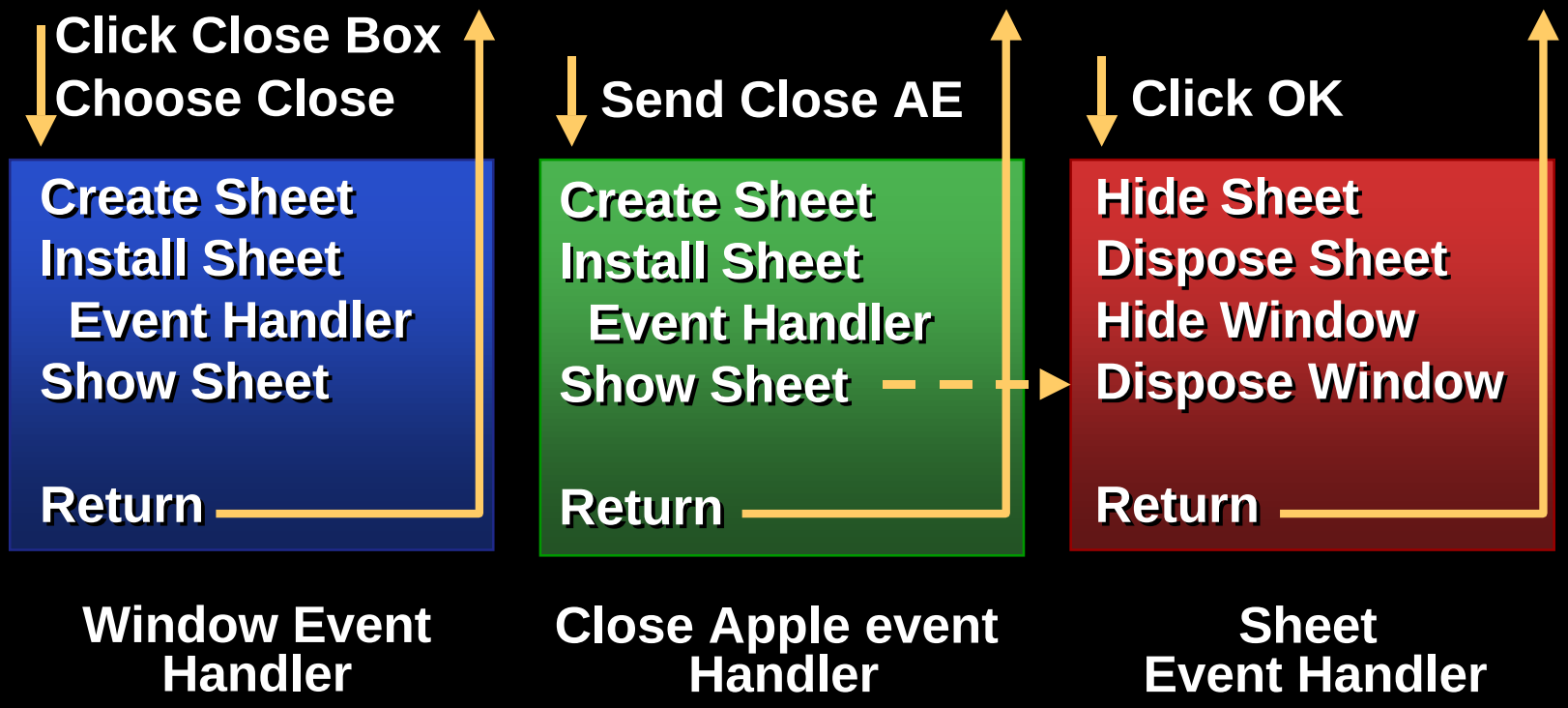
Handling Sheets

- Window-modal dialog boxes
- Initiate, but don't complete, an action
- Use callback procedures to notify your application when sheet is dismissed

Handling Sheets: The Problem



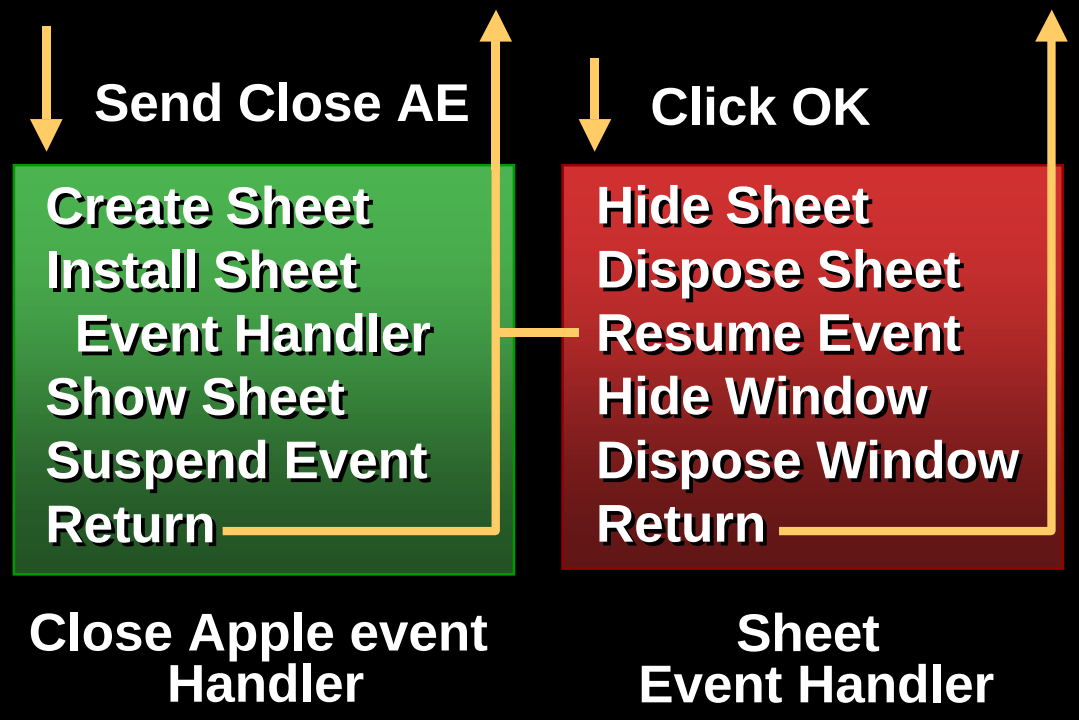
Handling Sheets: The Problem



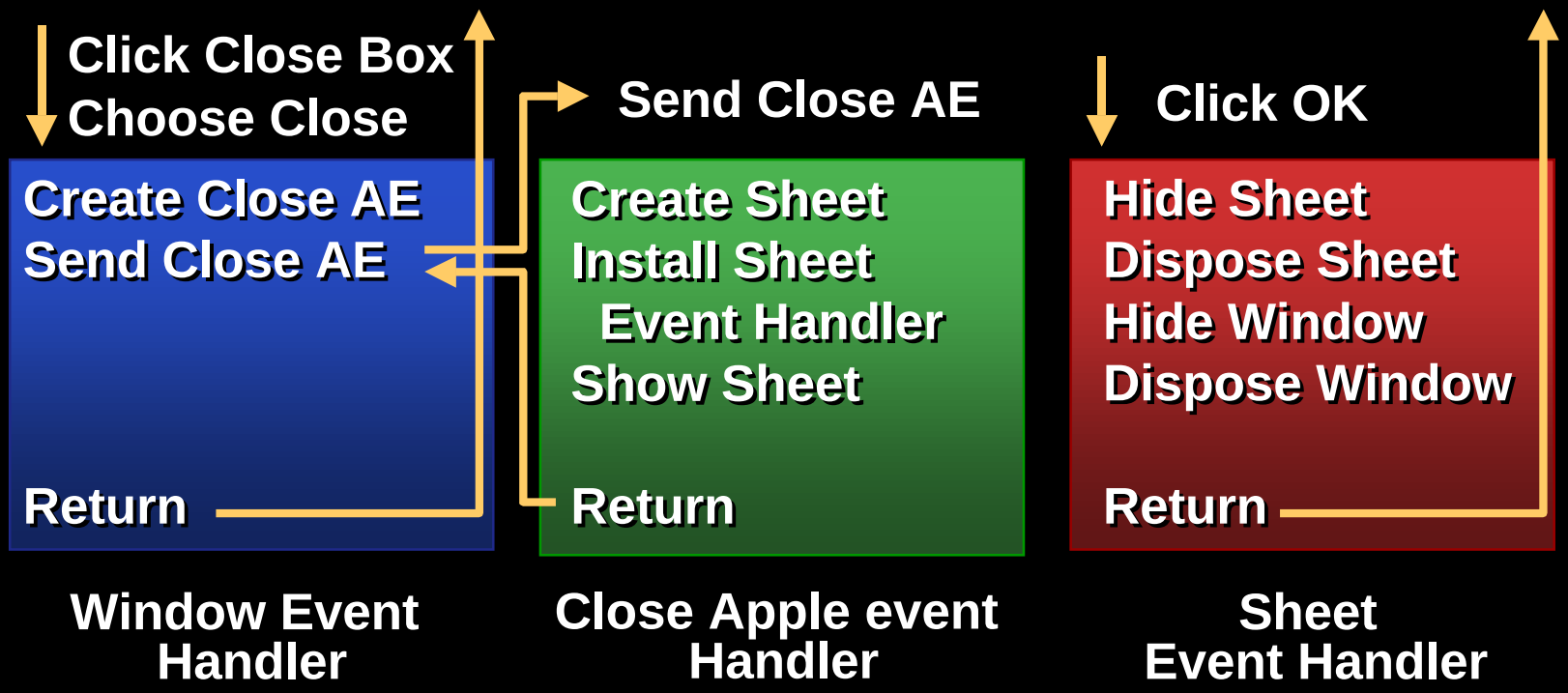
Handling Sheets: Solutions

- The easiest way: let the scripter skip the sheet!
- The wrong way: return from event after calling ShowSheetWindow() or NavDialogRun()
- The cheap way: make the dialog application-modal if invoked from an Apple event
- The nice way: suspend the event and resume it in the sheet event handler or Nav completion routine

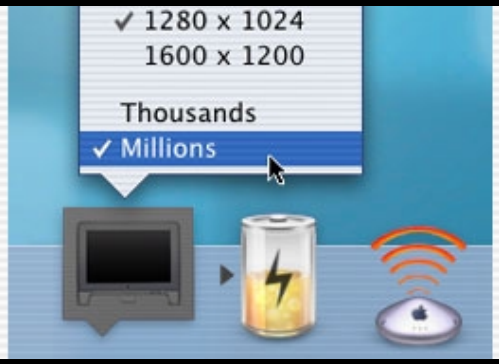
Handling Sheets: Solution



Handling Sheets: Solution



Demo



Handling Sheets: Issues

- Don’t resume sent-to-self events
- Don’t propagate AE errors out through your Carbon event handlers
  - Carbon will kindly redispatch the event

Handling Sheets: Issues

- Attachable applications
  - You want attached scripts to behave like external ones, but the events are send-to-self!
  - Always execute scripts on a separate thread
  - Write a Send function that blocks the thread
  - Unblock the thread in your sheet handler

Handling Sheets

- What this means to you
  - If you adopt window-modal dialogs, you have to change your event handlers
  - You pretty much have to go to the Carbon event model
  - Plan the relationship between your Carbon event handlers and their Apple event equivalents

Scripting Additions in Mac OS X

- Not loaded into the system heap at boot!
- Not loaded into your app's System table at launch!
- Loaded *only* when your application initializes the AppleScript component…
- …Or when you do the following magic stuff:

Loading Scripting Additions in Mac OS X

```
f = CFBundleGetFunctionPointerForName (
        CFBundleGetBundleWithIdentifier(
            CFSTR("com.apple.openscripting")),
    CFSTR("_OSAInstallStandardHandlers");
if (f) f();
err = AEBuildAppleEvent( 'ascr', 'gdut',
        typeProcessID, &selfPSN, sizeof(selfPSN),
        kAutoGenerateReturnID, kAnyTransactionID,
        &e,  &buildErr, "");
err = AESend(&e, &r, kAENoReply,
        kAENormalPriority, kAEDefaultTimeout,
        NULL, NULL);
```

Loading Scripting Additions in Mac OS X

- In the future, scripting additions may be loaded into every application
  - Loaded on demand, not at launch
  - **OSAInstallStandardHandlers** will go away
  - The code on the previous slide will Do The Right Thing

Debugging AppleScripts

- Using gdb
- Using third-party tools
- New debugger APIs

Debugging Using gdb

- **break UASExecute1**
  - Breaks before execution of any script or handler
- **break BCGetOperation()**
  - Breaks before every bytecode
  - Lets you single-step through a handler
- **break ComponentSend**
  - Breaks before event send
  - r3 is the outgoing event

Debugging Using Tools

- Script Debugger 3.0

Demo

Mark Alldritt
Late Night Software

Debugging APIs

- Coming soon
- Basic concepts:
  - Debug sessions (based on OSAIDs)
  - Stack frames (one per depth of handler call)
- Access to globals, locals, properties of handler
- Break, single step, step into, step over, continue

Debugging Via Apple Events

- Send an Apple event to any script to halt
- Other events parallel the debugging API
- Useful for cross-process debugging

Summary

- You can and should add scriptability **now!**
- It's easy to invoke scripts from your code
  - And if you do, be component-agnostic
- Migrate from FSSpecs
- Adopt Sheets
- Use the tools!

Roadmap

FF115 AppleScript Feedback Forum
Let us know how we're doing

Room C
5:00pm

Who to Contact

Jason Yeo
Mac OS Technology Manager
Apple Worldwide Developer Relations
jason@apple.com

Chris Espinosa
Manager, Components & Scripting
Apple Software Engineering
cde@apple.com

George Warner
AppleScript/Apple Events Support
Apple Developer Technical Support
geowar@apple.com

Q&A

With the AppleScript Engineering Team

http://developer.apple.com/events/urls.html

WWDC
Worldwide Developers Conference 2001